IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Patent Application No. 09/954,508

Applicant: Todorov et al.

Filed: September 14, 2001

TC/AU: 2144

Examiner: Nguyen, Thanh T.

Docket No.: 211626 (Client Reference No. 02,214 US)

Customer No.: 23460

APPELLANTS' APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In support of the appeal from the final rejection dated August 9, 2006, Appellants now submit their Brief.

Real Party In Interest

The patent application that is the subject of this appeal is assigned to Invensys Systems, Inc.

Related Appeals and Interferences

There are no appeals or interferences that are related to this appeal.

Status of Claims

Claims 1-50 stand finally rejected, and these rejections are presently being appealed.

A complete listing of these claims appears in the Claims Appendix.

Status of Amendments

There were no amendments submitted after the final rejection.

Summary of Claimed Subject Matter

Claims 1-50, including independent claims 1, 22 and 44 are pending. The summaries of the independent claims reference the specification and drawings filed with the application on September 14, 2001.

The invention recited in claim 1 supports multiple client data exchange protocols in a single data access server. In particular, the claimed invention is directed to a data access server (see, e.g., FIG. 3 and Data Access Server 50 in FIGs. 1 and 2) providing access to process data via a plurality of data exchange protocols (e.g., DDE, OPC and SuiteLink) supported by a plurality of client data exchange protocol modules (e.g., FIG. 3 Plugins 84, 86 and 88).

The process data access server recited in claim 1 includes a number of distinct functional components (see, FIG. 3 and page 10, line 19 to page 11, line 2). A device protocol interface (device protocol 96) facilitates accessing process data storage locations within the process control system. In the illustrative example, the device protocol interface delivers data received from data sources such as field devices and control processors to a data access server engine (DAS engine 90). The data access server recited in the claims includes *a set of client data exchange protocol modules* (plugins 84, 86 and 88 described at page 11, lines 3-22) providing access to data via particular *client data exchange protocols* (e.g., DDE, OPC and SuiteLink). The claimed data access server also comprises a server engine (DAS engine 90) for executing process data access requests received from clients via the client data exchange protocol modules. The server engine includes a *client application data exchange protocol abstraction layer* (standard interfaces 82 described at page 11, lines 16-19) comprising a set of operations callable by ones of the *set of client data exchange protocol modules* (e.g., plugins 84, 86 and 88) in response to receipt of requests from client applications (clients 60, 62 and 64 in FIG. 2 described at page 10, lines 1-17). The recited *abstraction layer* (standard interfaces 82 of the DAS engine 90) is interposed between the functional components of the data access server engine, which carry out requests, and the client data exchange modules (DDE plugin 84, OPC plugin 86, and SL plugin

88) that convert protocol-specific requests into the generic requests handled by the protocol abstraction layer (standard interfaces 82). The data exchange module-to-abstraction layer interface enables the data access server engine 90 to respond to requests submitted by client applications to the set of client data exchange modules according to multiple data exchange protocols.

The claimed modular approach for supporting particular data exchange protocols facilitates extending the supported set of client data exchange protocols. The presently claimed invention offers a new degree of extensibility to client application interfaces in a process control system. Enhanced extensibility/flexibility is achieved in a data access server by decoupling data access server engine functionality (carried out by a generic set of operations provided via interface 82) from the client data exchange protocols used by client applications to access process data via the data access server. The present invention achieves such decoupling by carrying out specific data exchange protocols in a set of program *modules*. These program modules are installed on the data access server to facilitate retrieval/presentation of data to the client applications according to a variety of protocols (e.g., DDE, OPC, SuiteLink, etc.) utilized by the client applications. After installation, the program modules provide a protocol-specific interface to client applications and communicate with the data access server engine via a standardized universal set of operations/interfaces 82. Extending the functionality of the data access server to support additional data exchange protocols is thus accomplished by providing and installing a new data exchange protocol module on the data access server. Previously existing software on the data access server, including the data access server engine and the previously installed protocol-specific protocol modules, need not be modified to include the new data exchange protocol module in the data access server system.

Independent **claim 22** recites a method for providing, by a data access server (50), access to process data in a distributed process control environment in accordance with a client application data exchange protocol supported by one of a set of client application data exchange protocol modules installed on the data access server. The set of client application data exchange protocol modules invoke a set of data access operations executable by a data access server engine (90) of the data access server (50) according to a module-engine interface definition (standard interfaces 82). The method, embodied in the illustrative example depicted in FIG. 7's sequence diagram (described at page 20, line 28 to page 22, line 3) includes initially receiving, by a first client application data exchange protocol module (e.g., OPC plugin) of the data access server, a first client application (OPC client) data access request ("AddItems" step 410) according to a first data exchange protocol (OPC).

Thereafter, the first client application data exchange protocol module (e.g., OPC plugin) generates a first data access operation call ("AddItems" step 412) for the data access server engine (90) conforming to the module-engine interface definition (interfaces 82), wherein the first data access operation call corresponds to the first client application data access request. In response the data access server engine (90) executes the first data access operation call ("Obtain data for items of interest" step 414).

Independent **claim 44**, corresponding to the illustrative example set forth in FIG. 5, is directed to a method for activating a data access server (50) through a start-up process that builds the data access server from previously installed program files including at least an executable file incorporating a data access server engine (90) and a separate and distinct file containing one or more of a set of client application data exchange protocol modules (plugins 84, 86, and 88) installed on the data access server (50). The set of client application data exchange protocol modules (plugins 84, 86 and 88) invoke a set of data access operations executable by the data access server engine (90) of the data access server (50) according to a module-engine interface definition (standard interfaces 82). The claimed method includes the step of starting up an executable corresponding to the data access server and including the data access server engine (step 206 described at page 18, lines 1-13). The method further comprises loading the set of client application data exchange protocol modules (e.g., OPC plugin 86) thereby creating program links between at least one of the protocol modules and the data access server executable (steps 214 and 216 described at page 18, lines 22-29). The method further comprises instantiating a data access server object corresponding to a connection between the data access server and a requesting client application (step 218 described at page 19, lines 1-12). Thus a specific connection is established, by a data access server object, for each particular client seeking access to the data access server's services.

Grounds of Rejection to be reviewed on Appeal

The grounds of rejection to be reviewed on appeal are the grounds stated in the Final Office Action mailed on August 9, 2006. In particular, Appellants appeal the rejection of Claims 1-50 as obvious under 35 U.S.C. Section 103(a) over Dorrance et al., U.S. Patent No. 6,430,598 (Dorrance) in view of Lim et al., U.S. Patent No. 6,718,550 (Lim).

Argument

Appellants request reversal of the rejection of presently pending claims 1-50 (provided in the Claims Appendix attached hereto) that are directed to a process data access server supporting multiple differing data exchange protocols through a set of client data exchange protocol modules that convert client protocol-specific requests into calls to a data exchange protocol abstraction layer of a data access server engine.

The claimed data access server provides access to process data via a plurality of *client data exchange protocols* supported by a *plurality of client data exchange protocol modules* (note plural form of terms). Furthermore, the data access server comprises a server engine that includes a *client application data exchange protocol abstraction layer* comprising a set of operations callable by ones of the *set of client data exchange protocol modules* in response to receipt of requests from client applications. The recited *abstraction layer* (standard interfaces 82 of the DAS engine 90) is interposed between the functional components of the data access server engine, which carry out requests, and the client data exchange modules (DDE plugin 84, OPC plugin 86, and SL plugin 88) that convert protocol-specific requests into the generic requests handled by the protocol abstraction layer.

This modular arrangement for supporting an extensible set of data exchange protocols in a process control system environment is not even remotely contemplated by the combined teachings of Dorrance and Lim. Dorrance discloses an email server that supports multiple email protocol requests through a single converter 65 entity. Lim does indeed disclose a set of clients. However, the presence of a set of clients does not overcome a complete absence of any teaching in either Dorrance or Lim with regard to a process data access server component arrangement wherein a set of protocol-specific client modules receive protocol-specific client requests and issue corresponding operation calls supported by a protocol abstraction layer of a data access server engine.

The grounds for Appellants' appeal are addressed further herein below.

Rejection of Claims 1-50 over Dorrance In View of Lim

Claims 1, 10-12, 22 and 32-34

a. Dorrance Does Not Disclose or Suggest The Claimed Set of Client Data Exchange Protocol Modules and Abstraction Layer

Appellants appeal the rejection of **claim 1** as being obvious over Dorrance in view of Lim because the Dorrance and Lim patents do not disclose, in combination, all of the elements of claim 1. The Dorrance patent, upon which the Final Office Action primarily relies, discloses an email server that supports message requests issued by a single client in multiple protocols. The multiple protocols are supported by a *single* converter 65 (see, FIG. 3). The converter 65 is a single integral component of the email server 62. The converter 65 receives message requests from a single client in potentially many different protocols (col. 6, lines 44-48). Thereafter, the converter 65 translates the received client requests into a standard server protocol (col. 6, lines 18-21). The Dorrance patent discloses handling multiple protocols via a single client interface component that handles all requests from clients regardless of their protocol. Furthermore, the Dorrance patent does not suggest that use of a single converter 65 within the email server 62 is somehow disadvantageous.

In contrast to claim 1, the converter 65 disclosed in Dorrance is not a module, of *a set of client data exchange protocol modules*, that submits calls to a set of operations supported by a client application data exchange protocol abstraction layer of a data access server engine. The Dorrance patent is silent as to the nature of the relationship between the converter 65 and other components of email server 62. However, the fact that the converter 65 handles multiple protocols as well as the complete absence of any description of a modular (or extensible) design for the email server 62 suggests that the converter 65 is an integral component of the email server 62.

The Final Office Action asserts that the Lim patent discloses a set of clients. Appellants agree that the Lim patent does indeed disclose a set of clients. However, the Lim patent is completely silent with regard to supporting multiple client data exchange protocols through the use of a set of client data exchange protocol modules.

Dorrance is deficient in the specific point where the invention recited in claim 1 departs from the prior art – supporting multiple data exchange protocols through a set of data exchange protocol *modules*. Furthermore, Appellants assert that there is no disclosure of Appellants' disclosed and claimed *data exchange protocol abstraction layer comprising a set of operations callable by ones of the set of client data exchange protocol modules*. The Office Action cites the server 62 *and* converter 65 in support of its asserted teaching of the "abstraction layer" in the Dorrance patent. By combining the server 62 and converter 65 together, the Final Office Action is, in effect, admitting that Dorrance merely discloses a single "black box" that handles requests from a client in a variety of protocols. However, there are clearly no specific teaching within Dorrance of Appellants' claimed *set of client data exchange protocol modules* and a *protocol abstraction layer* (within a data access server engine) including a set of operations callable by the protocol modules.

The Final Office Action concedes that Dorrance does not teach a "set of client" data exchange protocol modules. However, the Final Office Action does not appear to appreciate the functionality or purpose of Appellants' recited "set of client data exchange protocol modules" or the corresponding "abstraction layer" with which the multiple client data exchange protocol modules interface. Instead, the Office Action references passages in Lim relating to the presence of multiple "clients" – not client data exchange protocol modules.

Appellants furthermore traverse the Final Office Action's stated basis for combining the Dorrance and Lim references. The Final Office Action states that combining Lim's multiple clients with Dorrance "would have provided specific functions that can improve and reduce the performance of object in distributed object system". To the extent this statement is understood, it asserts that combining Lim with Dorrance somehow would improve the operation of Dorrance. However, operationally, Dorrance does not seem to have any operational shortcomings for its intended use in an email server system.

Appellants respectfully submit that the recited invention in claim 1 is not rendered obvious by the combined teachings of the Dorrance and Lim patents. The Dorrance patent neither discloses nor suggests a need to modularize client request protocol interfaces which, in turn, communicate with a data access server engine via an abstraction layer. The Lim patent merely discloses multiple clients. The email system disclosed in the Dorrance patent can already handle multiple client requests using multiple different protocols. The claimed invention is not rendered obvious to one skilled in the art by the combined teachings of Dorrance and Lim.

The Dorrance patent discloses an email server system that does not appear to need the claimed invention, including a set of modular client data exchange protocol-specific components and an abstraction layer on a data access server engine that provides an interface for the components. Appellants respectfully request an explanation of the stated grounds/motivation for combining the teachings of Dorrance and Lim to render the claimed invention at the bottom of page 3 of the Office Action. Appellants note that neither Dorrance nor Lim appears to disclose Appellants' recited *modules*. Appellants furthermore request specific identification of teachings of the recited "abstraction layer" in Dorrance – as opposed to the present general reference to the entire email server 62 and converter 65 which is provided in the pending Office Action.

b. Dorrance Does Not Even Disclose a "Process Data Access Server"

Appellants further note that the Dorrance patent is directed to an email server and does not constitute a data access server for process control systems. Such control systems comprise a wide variety of discrete and distributed regulatory control systems. However, even given its broadest interpretation, control systems would not appear to include email servers of the type disclosed in Dorrance. In the event the Final Office Action is not withdrawn, Appellants request a statement regarding the relationship between the Dorrance system and the claimed invention that includes elements of a *process control system.*

Appellants appeal the rejection of independent **claim 22** for the reasons set forth above with regard to **claim 1. Claim 22** defines a method for responding to client requests by one of a set of client data exchange protocol modules that is neither disclosed nor suggested by the combined teachings of the Dorrance and Lim patents.

Claims 2-5, 30, and 45-48

Appellants appeal the rejection of **claims 2-5, 30, and 45-48** for at least the reasons set forth herein with regard to claims 1, 22 and 44. Furthermore, Appellants submit that a *prima facie* obviousness rejection has not been presented since neither Dorrance nor Lim disclose plugins – or any form of modular software components. In the event that the final rejection is not withdrawn, Appellants request identification of a teaching of plugins within either Dorrance or Lim.

Claims 6, 31 and 49

Appellants appeal the rejection of **claims 6, 31 and 49** for at least the reasons set forth herein with regard to the claims from which they depend. In particular, neither Dorrance nor Lim discloses a modular approach for supporting client data exchange protocols. In the case of Dorrance, a single component handles all supported protocols. Lim does not include any disclosure that would suggest providing a set of protocol-specific modules.

Claim 7

In addition to the grounds recited above for claim 1, Appellants appeal the rejection of dependent **claim 7** for at least the additional reason that the Office Action has not identified any module "loading" mechanism as recited in claim 7, and instead merely references FIG. 3 of Dorrance which does not even identify a startup process or module loading function. In the event the rejection is not withdrawn, Appellants request identification of the specific portions of FIG. 3 in Dorrance that teach the claimed startup process and protocol module loading mechanism.

Claims 8, 25, and 50

Appellants appeal the rejection of **claims 8, 25 and 50** for the reasons described with regard to the independent claims from which they depend. Furthermore, claims 8, 25 and 50 recite multiple differing types of data exchange protocol modules calling a same callable operation on the server engine. In the event that the rejection is not withdrawn, Appellants request identification of specific teachings in the Dorrance patent showing that two distinct protocol modules (not clients) call a same operation on the data access server engine. Since

Dorrance discloses a single converter 65 (asserted by the Office Action as corresponding to the protocol modules), such teaching cannot be present in Dorrance since at least two converters would have to be present.

Claim 9

Appellants appeal the rejection of **claim 9** for at least the above reasons described with regard to claim 1. The cited reference neither discloses nor suggests an arrangement wherein the client data exchange module and data access server comprise independently designated files. In the event the rejection is not withdrawn, Appellants request specific identification of the portions of the cited reference wherein the recited start-up process is carried out through independently designateable protocol module and data access server files.

Claims 13-21, 23, 24, 26-29 and 35-43

Appellants appeal the rejection of **claims 13-21, 23, 24, 26-29 and 35-43**. The rejected claims recite specific operations that are known generally in the data access server art. However, none have been incorporated into an interface supported by the recited data access server engine and callable by a variety of protocol-specific modules as recited in claims 1 and 22. None of these operations are disclosed in either Dorrance or Lim. *Appellants further note that the citations to the prior art do not appear to match the assertions of alleged teachings of recited claim elements.* Appellants request submission of proper citations to teachings in the prior art if the rejections are not withdrawn.

Claim 44

Claim 44 is an independent claim directed to the dynamic creation of the data access server defined in claim 1 that is neither disclosed nor even remotely suggested in Dorrance and Lim. Appellants appeal the final rejection of independent **claim 44** for the reasons set forth above with regard to **claim 1**. Furthermore, *Appellants specifically note that the citations to the Dorrance and Lim patents in the rejection of claim 44 have little, if any, relevance to the recited elements of claim 44.* Appellants request further explanation of the references to the cited prior art corresponding to claim 44's recited elements.

Conclusion

In summary, the present invention is distinguishable from the combined teachings of the cited references for a variety of reasons. Most importantly, the invention recited in the presently pending claims is directed to a process data access server that supports a variety of client data exchange protocols (e.g., DDE, OPC, SuiteLink, etc.) via a set of protocol modules. The multiple protocol modules interface with a data access server engine via an abstraction layer comprising a set of callable operations. While the prior art does indeed disclose supporting multiple protocols, the recited *way* in which multiple client support is provided in Appellants' claimed invention is neither disclosed nor suggested in the prior art. For at least the reasons set forth herein, each of the presently pending claims is patentable over the prior art.

Appellants therefore request reversal of the presently pending rejection of claims 1-50.

Respectfully submitted,

[illegible]

Mark Joy, Reg. No. 35,562
LEYDIG, VOIT & MAYER, LTD.
Two Prudential Plaza
180 North Stetson Ave., Suite 4900
Chicago, Illinois 60601-6731
(312) 616-5600 (telephone)
(312) 616-5700 (facsimile)

Date: June 29, 2007

Claims Appendix

1. (Original) A process data access server enabling client applications incorporating potentially multiple differing data exchange protocols to access process data stored at potentially many different locations in a process control system, the process data access server comprising:

a device protocol interface facilitating accessing process data storage locations within the process control system;

a set of client data exchange protocol modules enabling client applications to request access to process data storage locations via the process data access server according to particular client data exchange protocols supported by the set of client data exchange protocol modules; and

a data access server engine for executing process data access requests, received by the process data access server via the set of client data exchange protocol modules, by accessing, via the device protocol interface, data storage locations corresponding to the process data access requests, and wherein the data access server engine includes a client application data exchange protocol abstraction layer comprising a set of operations callable by ones of the set of client data exchange protocol modules in response to receipt by the set of client data exchange protocol modules of process data access requests.

2. (Original) The process data access server of claim 1 wherein the set of client data exchange protocol modules comprise plugins.

3. (Original) The process data access server of claim 2 wherein at least one of the set of client data exchange protocol plugins comprises a dynamic plugin.

4. (Original) The process data access server of claim 2 wherein at least one of the set of client data exchange protocol plugins comprises a static plugin.

5. (Original) The process data access server of claim 2 wherein the set of protocol conversion modules comprise both static and dynamic plugins.

6. (Original) The process data access server of claim 1 wherein ones of the set of client data exchange protocol modules handle data access requests from client applications in accordance with particular client data exchange protocols.

7. (Original) The process data access server of claim 1 further including:

a loading mechanism for determining a presence of at least one of the set of client data exchange protocol modules upon a machine for executing the process data access server, and loading the at least one client data exchange protocol module during a startup process that integrates the at least one client data exchange module with the data access server engine.

8. (Original) The process data access server of claim 1 wherein the set of operations of the data access server engine includes at least one operation callable by at least two distinct ones of the set of client data exchange protocol modules that incorporate distinct data exchange protocols.

9. (Original) The process data access server of claim 1 wherein an operational data access server including the device protocol interface, the set of client data exchange protocol modules, and the data access server is created by a start-up process that builds the operational data access server from previously installed program files, and wherein the program files of the client data exchange protocol modules and the data access server are independently designateable with regard of one another.

10. (Original) The process data access server of claim 1 wherein the set of interface operations executable by the data access server engine includes an asynchronous data read operation for providing data from an identified data source in response to a client application data request.

11. (Original) The process data access server of claim 1 wherein the set of interface operations executable by the data access server engine includes a synchronous read operation that, in accordance with a timer duration expiration event, updates identified process data values via the device protocol interface.

12. (Original) The process data access server of claim 11 wherein the synchronous read operation discards an updated process data value for a data item that is determined to be unchanged from a current stored value for the data item, thereby avoiding transmissions of unchanged data values between the process data access server and requesting client applications.

13. (Original) The process data access server of claim 1 wherein the set of interface operations executable by the data access server engine includes a group creation operation that creates a first logical group containing a first set of data items.

14. (Original) The process data access server of claim 13 wherein a second logical group containing a second set of data items is includable as an item within the first logical group containing the first set of data items.

15. (Original) The process data access server of claim 13 wherein the set of interface operations executable by the data access server engine includes a group remove operation that removes a specified group from the process data access server.

16. (Original) The process data access server of claim 13 wherein the set of interface operations executable by the data access server engine includes operations for modifying the contents of the first logical group.

17. (Original) The process data access server of claim 1 wherein the set of interface operations executable by the data access server engine includes a write operation to a specified data item accessible by the process data access server.

18. (Original) The process data access server of claim 1 wherein the set of interface operations includes a data reference structure search operation that returns a data item reference corresponding to a data item value accessible by the client applications via the process data access server.

19. (Original) The process data access server of claim 18 wherein the data item reference is a handle.

20. (Original) The process data access server of claim 1 wherein the set of interface operations includes an error code generator that supplies error code text to a requesting client data exchange protocol module.

21. (Original) The process data access server of claim 1 wherein the set of interface operations includes a status reporter operation that provides access to a data structure that stores status values for the process data access server.

22. (Original) A method for providing, by a data access server, access to process data in a distributed process control environment in accordance with a client application data exchange protocol supported by one of a set of client application data exchange protocol modules installed on the data access server, and wherein the set of client application data exchange protocol modules invoke a set of data access operations executable by a data access server engine of the data access server according to a module-engine interface definition, the method comprising the steps of:

receiving, by a first client application data exchange protocol module of the data access server, a first client application data access request according to a first data exchange protocol;

first generating, by the first client application data exchange protocol module, a first data access operation call for the data access server engine conforming to the module-engine interface definition, wherein the first data access operation call corresponds to the first client application data access request; and

executing, by the data access server engine, the first data access operation call.

23. (Original) The method of claim 22 further comprising the steps of:

second generating, by the data access server engine, a response to the first data access operation call; and

third generating, by the first data exchange protocol module, a response to the first client application data access request, wherein the response to the first client application data access request corresponds to the response to the first data access operation call generated by the data access server engine during the second generating step.

24. (Original) The method of claim 22 further comprising the steps of:

second receiving, by a second client application data exchange protocol module of the data access server, a second client application data access request according to a second data exchange protocol; and

second generating, by the second client application data exchange protocol module, a second data access operation call for the data access server engine conforming to the module-engine interface definition, wherein the second data access operation call corresponds to the second client application data access request.

25. (Original) The method of claim 24 wherein the first data access operation call is identical to the second data access operation call.

26. (Original) The method of claim 22 further comprising the step of:

receiving, by the first client application data exchange protocol module, a request to create a logical group that contains a set of data items representing data accessed in the process control environment, and a further request to add a data item to the logical group.

27. (Original) The method of claim 26 wherein the first client application data access request comprises a subscription query requesting the data access server to issue a notification in response to detecting a change to a data value associated with the data item within the logical group.

28. (Original) The method of claim 27 wherein the executing step comprises forwarding a request for device data to a device protocol interface, and wherein the device protocol interface transmits a corresponding data request to a field device according to a field device-specific request protocol.

29. (Original) The method of claim 28 further comprising the steps of:

receiving, by the device protocol interface, a response from the field device comprising data corresponding to the data item;

forwarding, by the device protocol interface to the data access server engine, a response message including a data value for the data item.

30. (Original) The method of claim 22 wherein the set of client data exchange protocol modules comprise plugins.

31. (Original) The method of claim 22 wherein ones of the set of client data exchange protocol modules handle data access requests from client applications in accordance with particular client data exchange protocols.

32. (Original) The method of claim 22 wherein the executing step comprises performing an asynchronous data read operation for providing data from an identified data source in response to a client application data request.

33. (Original) The method of claim 22 wherein the executing step comprises performing a synchronous read operation that, in accordance with a timer duration expiration event, updates identified process data values via the device protocol interface.

34. (Original) The method of claim 33 wherein the synchronous read operation discards an updated process data value for a data item that is determined to be unchanged from a current stored value for the data item, thereby avoiding transmissions of unchanged data values between the process data access server and requesting client applications.

35. (Original) The method of claim 22 wherein the first data access operation call comprises a group creation operation that creates a first group containing a first set of data items.

36. (Original) The method of claim 35 further comprising executing, by the data access server engine, a second data access operation call that adds a second logical group containing a second set of data items as a group item within the first group.

37. (Original) The method of claim 35 further comprising executing, by the data access server engine, a second data access operation call that removes a specified group from the data access server.

38. (Original) The method of claim 35 further comprising executing, by the data access server engine, a second data access operation call to modify contents of the first logical group.

39. (Original) The method of claim 22 wherein the first data access operation call comprises a write operation to a specified data item accessible by the data access server.

40. (Original) The method of claim 22 wherein the first data access operation call comprises a data reference structure search operation that returns a data item reference corresponding to a data item value accessible by the client applications via the process data access server.

41. (Original) The method of claim 40 wherein the data item reference is a handle.

42. (Original) The method of claim 22 wherein the first data access operation call comprises an error code generator operation that supplies error code text to the requesting client data exchange protocol module.

43. (Original) The method of claim 22 wherein the first data access operation call comprises a status reporter operation that provides access to a data structure that stores status values for the data access server.

44. (Original) A method for activating a data access server through a start-up process that builds the data access server from previously installed program files including at least an executable file incorporating a data access server engine and a separate and distinct file containing one or more of a set of client application data exchange protocol modules installed on the data access server, and wherein the set of client application data exchange protocol modules invoke a set of data access operations executable by the data access server engine of the data access server according to a module-engine interface definition, the method comprising the steps of:

starting up an executable corresponding to the data access server and including the data access server engine;

loading the set of client application data exchange protocol modules thereby creating program links between at least one of the protocol modules and the data access server executable; and

instantiating a data access server object corresponding to a connection between the data access server and a requesting client application.

45. (Original) The method of claim 44 wherein the set of client data exchange protocol modules comprise plugins.

46. (Original) The method of claim 45 wherein at least one of the set of client data exchange protocol plugins comprises a dynamic plugin.

47. (Original) The method of claim 45 wherein at least one of the set of client data exchange protocol plugins comprises a static plugin.

48. (Original) The method of claim 45 wherein the set of protocol conversion modules comprise both static and dynamic plugins.

49. (Original) The method of claim 44 wherein ones of the set of client data exchange protocol modules handle data access requests from client applications in accordance with particular client data exchange protocols.

50. (Original) The method of claim 44 wherein the set of operations of the data access server engine includes at least one data access operation callable by at least two distinct ones of the set of client data exchange protocol modules that incorporate distinct data exchange protocols.

Evidence Appendix
NOT APPLICABLE

Related Proceedings Appendix
NOT APPLICABLE